Đã bắt đầu vào	Thứ bảy, 19 Tháng mười một 2022, 10:17 AM
lúc	
Tình trạng	Đã hoàn thành
Hoàn thành vào	Thứ ba, 29 Tháng mười một 2022, 9:09 PM
lúc	
Thời gian thực	10 ngày 10 giờ
hiện	
Điểm	3,00/3,00
Điểm	10,00 của 10,00 (100 %)



Câu hỏi 1

Chính xác

Điểm 1,00 của 1,00

Given class SplayTree definition:



```
class SplayTree {
    struct Node {
        int val;
        Node* pLeft;
        Node* pRight;
        Node* pParent;
        Node(int val = 0, Node* 1 = nullptr, Node* r = nullptr, Node* par = nullptr) : val(val), pLeft(1), pRight(r), pParent(par)
{ }
    };
    Node* root;
    // print the tree structure for local testing
    void printBinaryTree(string prefix, const Node* root, bool isLeft, bool hasRightSibling) {
        if (!root && isLeft && hasRightSibling) {
            cout << prefix << "├─\n";
        if (!root) return;
        cout << prefix;</pre>
        if (isLeft && hasRightSibling)
            cout << "\—";
        else
            cout << "└─";
        cout << root->val << '\n';</pre>
        printBinaryTree(prefix + (isLeft && hasRightSibling ? "| " : " "), root->pLeft, true, root->pRight);
        printBinaryTree(prefix + (isLeft && hasRightSibling ? "| " : "
                                                                           "), root->pRight, false, root->pRight);
    }
    void printPreorder(Node* p) {
        if (!p) {
            return;
        cout << p->val << ' ';
        printPreorder(p->pLeft);
        printPreorder(p->pRight);
    }
public:
    SplayTree() {
        root = nullptr;
    ~SplayTree() {
        // Ignore deleting all nodes in the tree B \circ I + CMUT - CNCP
    }
    void printBinaryTree() {
        printBinaryTree("", root, false, false);
    }
    void printPreorder() {
        printPreorder(root);
        cout << "\n";</pre>
    }
    void splay(Node* p) {
        // To Do
    void insert(int val) {
        // To Do
    }
};
```

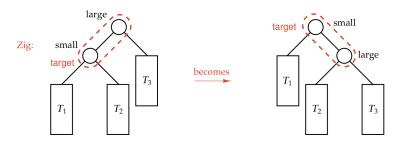
Implement the following method:

1. void splay(Node* p): bottom-up splaying a Node

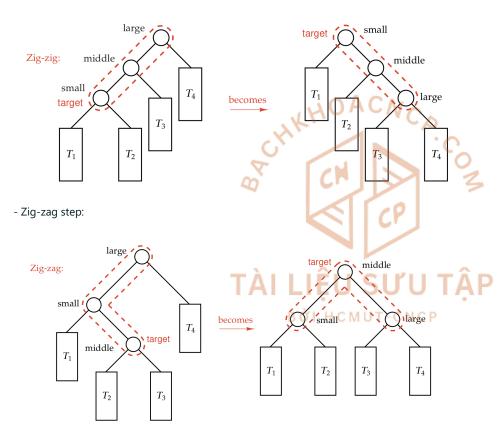
When a splay operation is performed on Node p, it will be moved to the root. To perform a splay operation we carry out a sequence of splay steps, each of which moves p closer to the root.

The three types of splay steps are:

- Zig step



- Zig-zig step:



Note: there are also zag, zag-zag and zag-zig step but we don't show them here

2. void insert(int val):

To insert a value val into a splay tree:

- + Insert val as with a normal binary search tree.
- + When the new value is inserted, a splay operation is performed. As a result, the newly inserted node becomes the root of the tree.

Note: In a splay tree, the values the in left subtree <= root's value <= the values in the right subtree. In this exercise, when inserting a duplicate value, you have to insert it to the right subtree to pass the testcases.

Constraint of testcases:

- + number of operation <= 10^4
- + 1 <= val <= 10^5

For example:

```
Test
                                           Input
                                                       Result
SplayTree tree;
                                                       90 80 30 70 50 100
int query;
                                           insert 50
                                                       L--90
cin >> query;
                                           insert 70
                                                             -80
for(int i = 0; i < query; i++) {</pre>
                                           insert 30
                                                                -30
                                           insert 80
    string op;
    int val;
                                           insert 100
                                                                    70
    cin >> op >> val;
                                           insert 90
    if (op == "insert")
                                                           L-100
        tree.insert(val);
}
// print preorder traversal of the tree
tree.printPreorder();
// print structure of the tree
tree.printBinaryTree();
SplayTree tree;
                                                       95 95 80 200 100 200
int query;
                                           insert 95
                                                       L-95
cin >> query;
                                           insert 200
                                                             -95
                                                             L--80
for(int i = 0; i < query; i++) {</pre>
                                           insert 80
    string op;
                                           insert 100
                                                             -200
    int val;
                                           insert 200
                                                                -100
    cin >> op >> val;
                                           insert 95
    if (op == "insert")
        tree.insert(val);
// print preorder traversal of the tree
tree.printPreorder();
// print structure of the tree
tree.printBinaryTree();
```

Answer: (penalty regime: 0 %)

```
Reset answer
```

```
// write your helper functions here
 2
    void rotateRight(Node *node)
    {
 3 •
        Node *T=node->pLeft;
 4
                                            BÓI HCMUT-CNCP
 5
        Node *B=T->pRight;
        Node *D=node->pParent;
 6
        if(D)
 7
 8
 9
            if(D->pRight==node) D->pRight=T;
10
            else D->pLeft=T;
11
12
        if(B) {
13
            B->pParent=node;
14
15
        T->pParent=D;
16
        T->pRight=node;
17
18
        node->pParent =T;
        node->pLeft=B;
19
20
21
22
    void rotateLeft(Node *node)
23
24
            Node *T=node->pRight;
25
26
        Node *B=T->pLeft;
        Node *D=node->pParent;
27
        if(D)
28
29
            if(D->pRight==node) D->pRight=T;
30
31
            else D->pLeft=T;
32
        if(B) {
33 •
```

```
35
36
        T->pParent=D;
37
        T->pLeft=node;
38
39
        node->pParent =T;
```

	Test	Input		Expected		Got		
/	SplayTree tree;	6		90 80 30 70	50 100	90 80 30 7	0 50 100	~
	int query;	insert	50	<u>└</u> 90		L90		
	<pre>cin >> query;</pre>	insert	70	 80		 80		
	for(int i = 0; i < query; i++) {	insert	30	└─30		L3	10	
	string op;	insert	80	I				
	int val;	insert	100		70	ι	 70	
	cin >> op >> val;	insert	90	I	L50		L50	
	if (op == "insert")			L-100		└──100		
	<pre>tree.insert(val);</pre>							
	}							
	// print preorder traversal of the tree							
	<pre>tree.printPreorder();</pre>							
	// print structure of the tree							
	<pre>tree.printBinaryTree();</pre>							
	Culay Tage trace		. 1	95 95 80 200	100 200	05 05 00 3	100 100 200	
	<pre>SplayTree tree;</pre>	6	7	_	100 200		100 100 200	•
	int query;	insert insert		L—95	- /~	95		
	cin >> query;			95	•		10	
	for(int i = 0; i < query; i++) {	insert		80			80	
	string op;	insert		└─ <u>200</u> └─ <u>100</u>		200	00	
	int val;	insert	1	100			.00	
	cin >> op >> val;	insert	95					
	if (op == "insert")				200	'	200	
	<pre>tree.insert(val);</pre>							
	}							
	// print preorder traversal of the tree							
	tree.printPreorder();	1.1	18	11 61	PII:	ΓÂD		
	// print structure of the tree			:U 5L	JU .	IAP		
	<pre>tree.printBinaryTree();</pre>							

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

10

Câu hỏi 2

Chính xác

Điểm 1,00 của 1,00

Given class SplayTree definition:

```
class SplayTree {
    struct Node {
        int val;
        Node* pLeft;
        Node* pRight;
        Node* pParent;
        Node* pParent;
        Node(int val = 0, Node* 1 = nullptr, Node* r = nullptr, Node* par = nullptr) : val(val), pLeft(1), pRight(r), pParent(par)
{ }
    };
```

Node* root;

```
// print the tree structure for local testing
void\ printBinaryTree(string\ prefix,\ const\ Node*\ root,\ bool\ is Left,\ bool\ has Right Sibling)\ \{
    if (!root && isLeft && hasRightSibling) {
        cout << prefix << "|--\n";
    }
    if (!root) return;
    cout << prefix;</pre>
    if (isLeft && hasRightSibling)
        cout << "|--";
    else
        cout << "└─";
    cout << root->val << '\n';</pre>
    printBinaryTree(prefix + (isLeft && hasRightSibling ? "
                                                                          "), root->pLeft, true, root->pRight);
    printBinaryTree(prefix + (isLeft && hasRightSibling ? "
                                                                           ), root->pRight, false, root->pRight);
```

```
void printPreorder(Node* p) {
   if (!p) {
      return;
   }
   cout << p->val << ' ';
   printPreorder(p->pLeft);
   printPreorder(p->pRight);
}
```

```
public:
    SplayTree() {
       root = nullptr;
    }
```

```
~SplayTree() {
    // Ignore deleting all nodes in the tree
}
```

```
void printBinaryTree() {
    printBinaryTree("", root, false, false);
}
```

```
void printPreorder() {
   printPreorder(root);
   cout << "\n";
}</pre>
```

};

```
void splay(Node* p);

void insert(int val);

bool search(int val) {
    // To Do
```

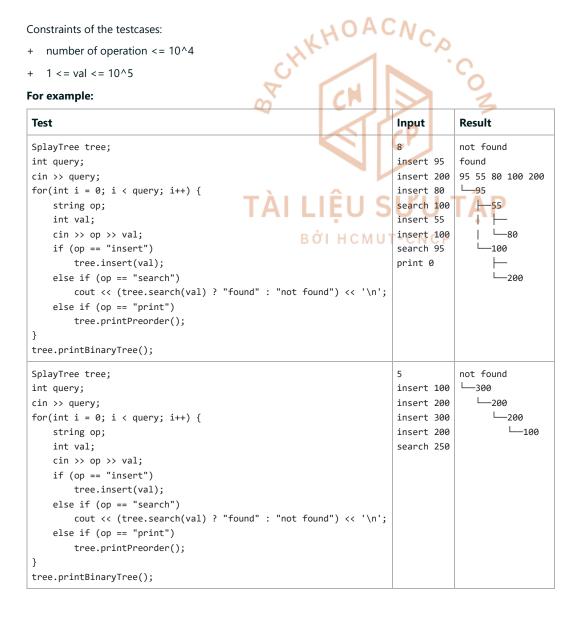
Method splay and insert are already implemented

You have to implement the following method:

bool search(int val): search for the value val in the tree.

The search operation in splay tree do the same thing as BST search. In addition, it also splays the node containing the value to the root.

- + If the search is successful, the node that is found will become the new root and the function return true.
- + Else, the last accessed node will be splayed and become the new root and the function return false.



Answer: (penalty regime: 0 %)

Reset answer

```
// Write your helper functions here
    bool searchRec(int val, Node*& node,Node*& par){
 2
        if(node == nullptr) {
 3
 4
            if(par) splay(par);
 5
            return false;
 6
 7
        if(val < node->val) return searchRec(val,node->pLeft,node);
        if(val > node->val) return searchRec(val,node->pRight,node);
 8
 9
        splay(node);
10
        return true;
11
    bool search(int val){
12
        if(root == nullptr) return false;
13
        return searchRec(val,root,root->pParent);
14
15
   }
```



	Test	Input	Expected	Got	
~	SplayTree tree; int query;	8 insert 95	not found	not found	~
	cin >> query;	insert 200	95 55 80 100 200	95 55 80 100 200	
	<pre>for(int i = 0; i < query; i++) { string op;</pre>	insert 80 search 100	└──95 ├──55		
	int val;	insert 55	i j-	i j-	
	<pre>cin >> op >> val; if (op == "insert")</pre>	insert 100 search 95			
	tree.insert(val);	print 0	<u> </u>	<u> </u>	
	<pre>else if (op == "search") cout << (tree.search(val) ? "found" : "not found") << '\n';</pre>		200	200	
	<pre>else if (op == "print") tree.printPreorder();</pre>				
	}				
	tree.printBinaryTree();				

	Test	Input		Expected	Got	
~	SplayTree tree;	5		not found	not found	~
	int query;	insert	100	<u></u> 300	<u></u> 300	
	cin >> query;	insert	200	L—200	<u></u> 200	
	for(int i = 0; i < query; i++) {	insert	300	<u></u> 200	L200	
	string op;	insert	200	└─100	<u>└</u> 100	
	int val;	search	250			
	cin >> op >> val;					
	<pre>if (op == "insert")</pre>					
	<pre>tree.insert(val);</pre>					
	else if (op == "search")					
	<pre>cout << (tree.search(val) ? "found" : "not found") << '\n';</pre>					
	else if (op == "print")					
	tree.printPreorder();					
	}					
	<pre>tree.printBinaryTree();</pre>					

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.



Câu hỏi 3

Chính xác

Điểm 1,00 của 1,00

Given class SplayTree definition:

```
class SplayTree {
    struct Node {
        int val;
        Node* pLeft;
        Node* pRight;
        Node* pParent;
        Node(int val = 0, Node* l = nullptr, Node* r = nullptr, Node* par = nullptr) : val(val), pLeft(l), pRight(r), pParent(par)
{ }
    };
    Node* root;
    // print the tree structure for local testing
    void printBinaryTree(string prefix, const Node* root, bool isLeft, bool hasRightSibling) {
        if (!root && isLeft && hasRightSibling) {
            cout << prefix << "-\n";
        }
        if (!root) return;
        cout << prefix;</pre>
        if (isLeft && hasRightSibling)
            cout << "\vdash—";
        else
            cout << "└─";
        cout << root->val << '\n';</pre>
        printBinaryTree(prefix + (isLeft && hasRightSibling ? "|
                                                                            "), root->pLeft, true, root->pRight);
                                                                      :/" [] "), root->pRight, false, root->pRight);
        printBinaryTree(prefix + (isLeft && hasRightSibling ? "|
```

```
void printPreorder(Node* p) {
    if (!p) {
        return;
    }
    cout << p->val << ' ';
    printPreorder(p->pLeft);
    printPreorder(p->pRight);
}
public:
    SplayTree() {
    root = nullptr;
}
```

```
~SplayTree() {
    // Ignore deleting all nodes in the tree
}
```

```
void printBinaryTree() {
    printBinaryTree("", root, false, false);
}
```

```
void printPreorder() {
    printPreorder(root);
    cout << "\n";
}</pre>
```

```
void splay(Node* p);
```

```
void insert(int val);
```

```
bool search(int val);
```

```
Node* remove(int val) {
    // To Do
  }
};
```

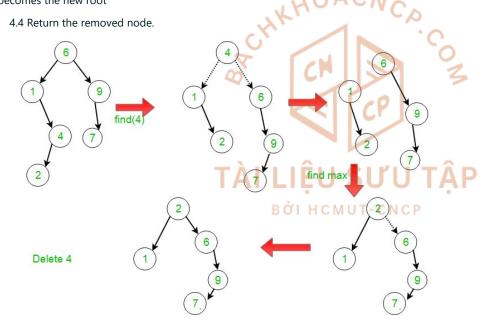
The methods splay, insert and search are already implemented.

Implement the following method:

Node* remove(int val): remove the first Node with value equal to val from the tree and return it.

To perform remove operation on splay tree:

- 1. If root is NULL, return the root
- 2. Search for the first node containing the given value val and splay it. If val is present, the found node will become the root. Else the last accessed leaf node becomes the root.
- 3. If new root's value is not equal to val, return NULL as val is not present.
- 4. Else the value val is present, we remove root from the tree by the following steps:
 - 4.1 Split the tree into two tree: tree1 = root's left subtree and tree2 = root's right subtree
 - 4.2 If tree1 is NULL, tree2 is the new root
- 4.3 Else, splay the leaf node with the largest value in tree1. tree1 will be a left skewed binary tree. Make tree2 the right subtree of tree1. tree1 becomes the new root



Constraints of the testcases:

- + number of operations <= 10^4
- + 1 <= val <= 10^5

For example:

t	Input	Result
SplayTree tree;	7	removed
int query;	insert 100	100 50 250 300
cin >> query;	insert 300	L—100
<pre>for(int i = 0; i < query; i++) {</pre>	insert 200	 50
string op;	insert 50	L—250
int val;	insert 250	<u> </u>
cin >> op >> val;	remove 200	300
<pre>if (op == "insert")</pre>	print 0	
<pre>tree.insert(val);</pre>		
else if (op == "remove")		
<pre>cout << (tree.remove(val) != nullptr ? "removed" : "not found") << '\n';</pre>		
else if (op == "search")		
<pre>cout << (tree.search(val) ? "found" : "not found") << '\n';</pre>		
else if (op == "print")		
<pre>tree.printPreorder();</pre>		
}		
tree.printBinaryTree();		
SplayTree tree;	7	not found
int query;	insert 900	100 750 800 900 14
cin >> query;	insert 1400	L—100
for(int i = 0; i < query; i++) {	insert 100	
string op;	insert 800	L750
int val;	insert 750	<u> </u>
<pre>int val; cin >> op >> val; if (op == "insert(") tree insert(val);</pre>	remove 500	L—800
if (op == "insert")	print 0	<u> </u>
tree.insert(val);		L—900
else if (op == "remove")		<u> </u>
<pre>cout << (tree.remove(val) != nullptr ? "removed" : "not found") << '\n';</pre>)	140
else if (op == "search")	2	
<pre>cout << (tree.search(val) ? "found" : "not found") << '\n';</pre>		
else if (op == "print")		
tree.printPreorder();		
}		

TÀI LIỆU SƯU TẬP

Answer: (penalty regime: 0. %)

BŐI HCMUT-CNCP

Reset answer

```
// Write your helper functions here
 1
 2
 3 •
   Node* remove(int val){
 4
 5
        if(root == nullptr) return nullptr;
        search(val);
 6
 7 ,
        if(root->val == val){
 8
            if(!root->pLeft){
 9
                Node* tmp = root;
10
                root = root->pRight;
11
                if(root) root->pParent = nullptr;
12
                return tmp;
13
14
            Node* treeL = root->pLeft;
            Node* treeR = root->pRight;
15
16
            treeL->pParent = nullptr;
17
            root->pLeft = root->pRight = nullptr;
            Node* ans = root;
18
19
            while(treeL->pRight) treeL = treeL->pRight;
20
            splay(treeL);
            treeL->pRight = treeR;
21
22
            if(treeR) treeR->pParent = treeL;
23
            return ans;
24
25
        return nullptr;
26
```

	Test	Input	Expected	Got	
,	SplayTree tree;	7	removed	removed	-
	int query;	insert	100 50 250 300	100 50 250 300	
	cin >> query;	100	L—100	L— ₁₀₀	
	for(int i = 0; i < query; i++) {	insert	 50	 50	
	string op;	300			
	int val;	insert		<u> </u>	
	cin >> op >> val;	200	300	L—300	
	if (op == "insert")	insert 50			
	tree.insert(val);	insert			
	else if (op == "remove")	250			
	cout << (tree.remove(val) != nullptr ? "removed" : "not	remove			
	found") << '\n';	200			
	else if (op == "search")	print 0			
	cout << (tree.search(val) ? "found" : "not found") <<	princ 0			
		2			
	'\n';				
	else if (op == "print")				
	tree.printPreorder();				
	}				
	<pre>tree.printBinaryTree();</pre>				1
	SplayTree tree;	7 ^	not found	not found	١,
	int query; TAI LIÊU SU'U	7 insert	100 750 800 900	100 750 800 900	
	cin >> query;	900	1400	1400	
	for(int i = 0; i < query; i++) {	insert	L—100	L—100	
	string op;	1400	 -	<u> </u>	
	int val;	insert	750	750	
	cin >> op >> val;	100		<u> </u>	
	<pre>if (op == "insert")</pre>	insert	L800	i800	
	tree.insert(val);	800	<u> </u>	<u> </u>	
	else if (op == "remove")	insert	900	L_900	
	cout << (tree.remove(val) != nullptr ? "removed" : "not	750	<u> </u>	<u> </u>	
	found") << '\n';	remove		'	
	else if (op == "search")	500	L—1400	L—1400	
	cout << (tree.search(val) ? "found" : "not found") <<	print 0	1400	1400	
	'\n';	pi ilic 0			
	else if (op == "print")				
	<pre>tree.printPreorder();</pre>				
	}				
	<pre>tree.printBinaryTree();</pre>				

Passed all tests! 🗸

Chính xác Điểm cho bài nộp này: 1,00/1,00.

BÁCH KHOA E-LEARNING



WEBSITE

HCMUT

MyBK

BKSI

LIÊN HÊ

- ♀ 268 Lý Thường Kiệt, P.14, Q.10, TP.HCM
- (028) 38 651 670 (028) 38 647 256 (Ext: 5258, 5234)
- elearning@hcmut.edu.vn

